

Práctica-(Tarea)

Verificación de la integridad de archivos mediante funciones hash



Alejandro Garcia Burguillos

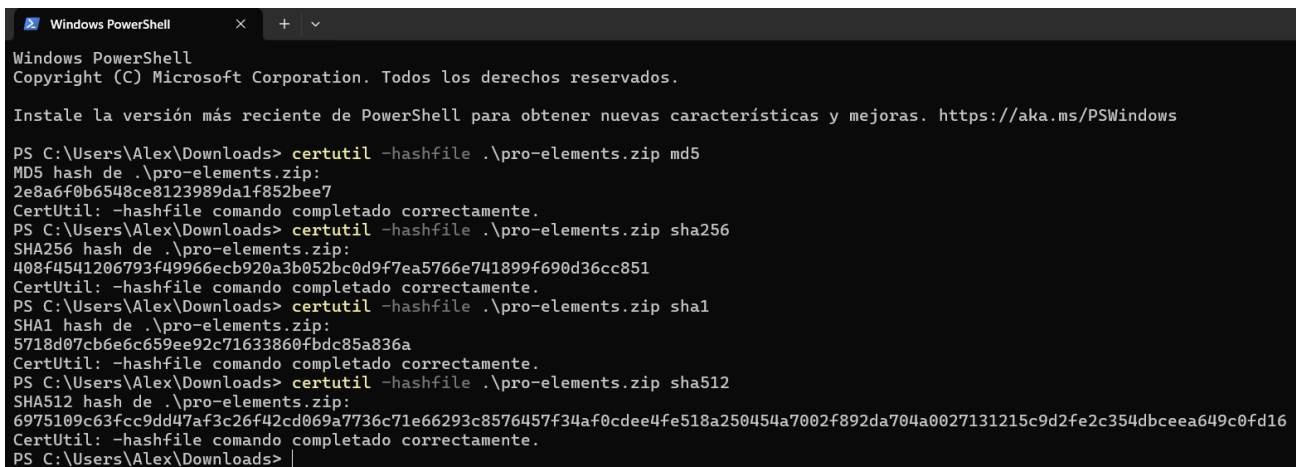
16/11/2024

1. Práctica en Windows:

- Utilizando la herramienta **CertUtil** en Windows, calcula el hash de un archivo que tengas disponible en tu ordenador (puede ser un archivo de texto pequeño o cualquier archivo que elijas).
 - Usa al menos dos algoritmos diferentes (ej. MD5 y SHA256).
 - Copia y pega en el informe los comandos utilizados y los resultados obtenidos.
 - Opcional: Instala **HashTab** o **QuickHash GUI** y genera el hash de un archivo usando uno de estos programas. Explica brevemente tu experiencia con estas herramientas (¿te parecieron fáciles de usar?, ¿cuál es la ventaja de tener una interfaz gráfica?).
- Haz captura del uso de la herramientas

Para la herramienta **CertUtil** tendremos que abrir **PowerShell**, aquí podremos usar el comando usando la siguiente estructura:

certutil -hashfile .\NuestroFichero AlgoritmoHash



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\Alex\Downloads> certutil -hashfile .\pro-elements.zip md5
MD5 hash de .\pro-elements.zip:
2e8a6f0b6548ce8123989da1f852bee7
CertUtil: -hashfile comando completado correctamente.
PS C:\Users\Alex\Downloads> certutil -hashfile .\pro-elements.zip sha256
SHA256 hash de .\pro-elements.zip:
408f4541206793f49966ecb920a3b052bc0d9f7ea5766e741899f690d36cc851
CertUtil: -hashfile comando completado correctamente.
PS C:\Users\Alex\Downloads> certutil -hashfile .\pro-elements.zip sha1
SHA1 hash de .\pro-elements.zip:
5718d07cb6e6c659ee92c71633860fbd85a836a
CertUtil: -hashfile comando completado correctamente.
PS C:\Users\Alex\Downloads> certutil -hashfile .\pro-elements.zip sha512
SHA512 hash de .\pro-elements.zip:
6975109c63fcc9dd47af3c26f42cd069a7736c71e66293c8576457f34af0cdee4fe518a250454a7002f892da704a0027131215c9d2fe2c354dbceea649c0fd16
CertUtil: -hashfile comando completado correctamente.
PS C:\Users\Alex\Downloads> |
```

Pero primero, ¿Qué es un hash?

Un hash es una huella digital única para un conjunto de datos. Es como un resumen corto que captura la esencia del contenido original, pero de una forma que es imposible de revertir. Los hashes se usan para verificar la integridad de archivos, asegurar contraseñas y validar firmas digitales, entre otras aplicaciones de seguridad.

Yo he probado con 4 algoritmos hash: md5, sha1, sha256 y sha512.

- **MD5** (Message Digest 5): Un algoritmo hash de 128 bits. Es relativamente antiguo y se ha descubierto que es vulnerable a colisiones, lo que significa que es posible encontrar dos conjuntos de datos diferentes que produzcan el mismo hash.
- **SHA-1** (Secure Hash Algorithm 1): Un algoritmo hash de 160 bits. Es más seguro que MD5, pero también tiene vulnerabilidades conocidas.
- **SHA-256** (Secure Hash Algorithm 256): Un algoritmo hash de 256 bits. Es uno de los algoritmos hash más utilizados en la actualidad y se considera bastante seguro.
- **SHA-512** (Secure Hash Algorithm 512): Un algoritmo hash de 512 bits. Es aún más seguro que SHA-256, pero requiere más tiempo para calcular.

Resultados obtenidos:

```
PS C:\Users\Alex\Downloads> certutil -hashfile .\pro-elements.zip md5
```

MD5 hash de .\pro-elements.zip:

2e8a6f0b6548ce8123989da1f852bee7

CertUtil: -hashfile comando completado correctamente.

```
PS C:\Users\Alex\Downloads> certutil -hashfile .\pro-elements.zip sha1
```

SHA1 hash de .\pro-elements.zip:

5718d07cb6e6c659ee92c71633860fdbc85a836a

CertUtil: -hashfile comando completado correctamente.

```
PS C:\Users\Alex\Downloads> certutil -hashfile .\pro-elements.zip sha256
```

SHA256 hash de .\pro-elements.zip:

408f4541206793f49966ecb920a3b052bc0d9f7ea5766e741899f690d36cc851

CertUtil: -hashfile comando completado correctamente.

```
PS C:\Users\Alex\Downloads> certutil -hashfile .\pro-elements.zip sha512
```

SHA512 hash de .\pro-elements.zip:

6975109c63fcc9dd47af3c26f42cd069a7736c71e66293c8576457f34af0cdee4fe518a2504

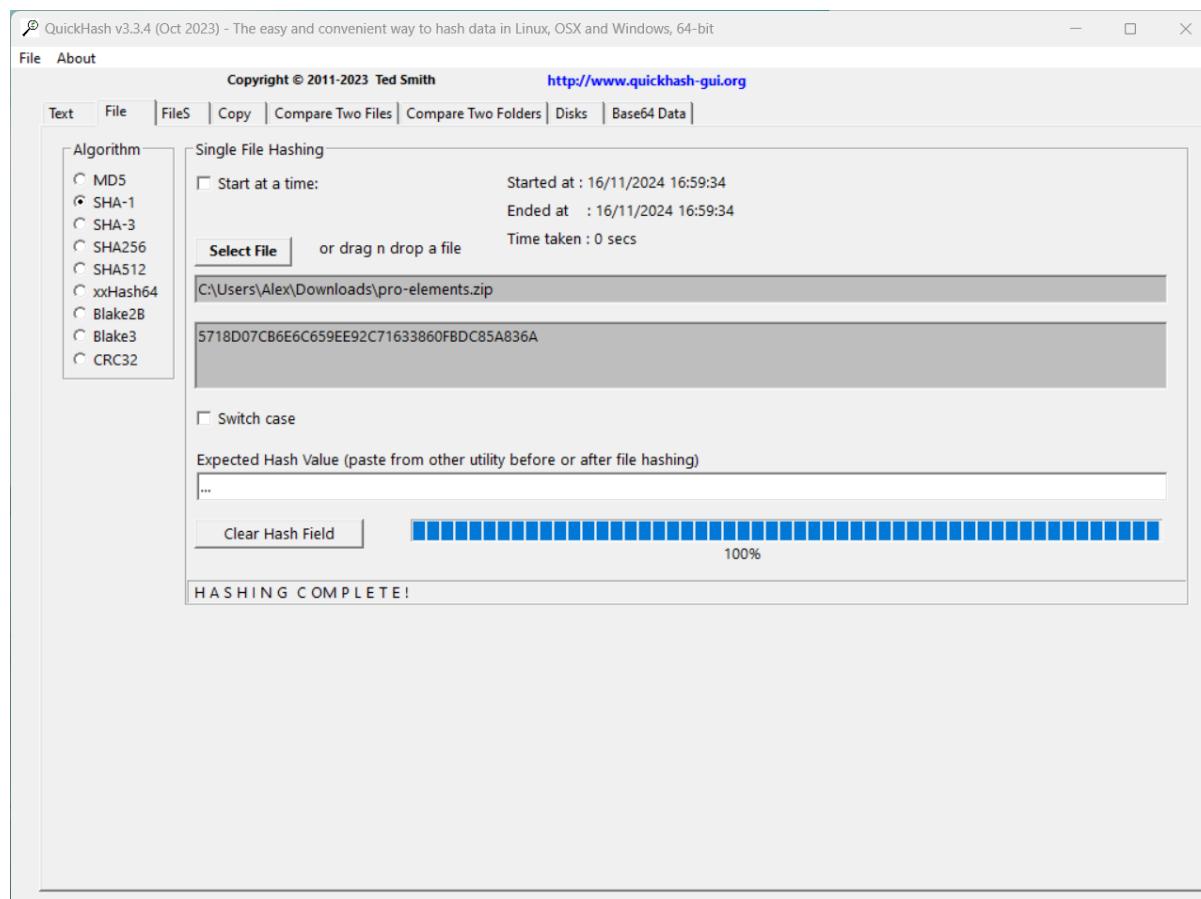
54a7002f892da704a0027131215c9d2fe2c354dbceea649c0fd16

CertUtil: -hashfile comando completado correctamente.

Opcional: Instalación de [QuickHash GUI](http://www.quickhash-gui.org).

La herramienta que voy a probar va a ser QuickHash, para usarlo descargamos el zip desde la página de [QuickHash](http://www.quickhash-gui.org) y abrimos el .exe que viene dentro.

Dentro del programa tendremos varias opciones, la que nos interesa está en file, donde elegiremos el mismo archivo comprobado antes en PowerShell



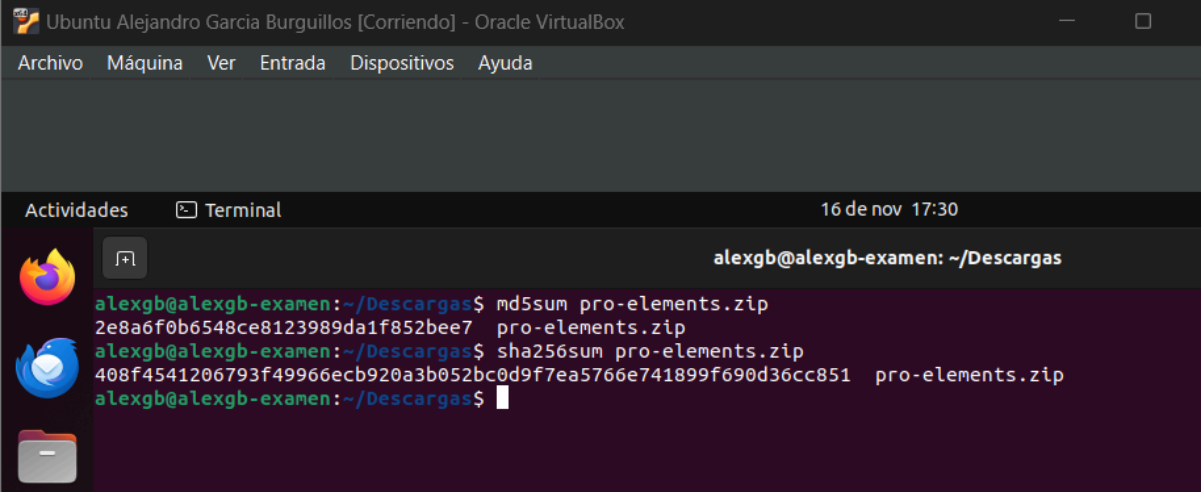
Comparando el resultado con el obtenido anteriormente con CertUtil vemos que es el mismo (En este caso utilizando el algoritmo **SHA-1**), como se esperaba, a parte de esto tendremos opciones más interesantes como la de comparar dos archivos o dos carpetas, esta es una muy buena opción para comprobar que un archivo no ha sido modificado o que el contenido que hemos descargado es original, si tenemos una copia que sabemos que es buena claro.

Añadir que el uso de **interfaz gráfica** hace que sea más **sencillo** de usar, si es algo que se va a usar a menudo merece la pena descargar la herramienta, además de tener opciones adicionales.

2. Práctica en Linux:

- Utilizando una máquina virtual con Linux o una distribución en modo live, calcula el hash de un archivo usando los siguientes comandos:
 - **md5sum** para generar el hash MD5.
 - **sha256sum** para generar el hash SHA-256.
 - Copia y pega en el informe los comandos utilizados y los resultados obtenidos.
- Explica cómo podrías utilizar estos comandos para verificar la integridad de un archivo descargado de internet.

Para la comprobación del hash en linux he descargado el mismo archivo utilizado en windows, y como se puede ver el hash coincide con los anteriores.



```
Ubuntu Alejandro Garcia Burguillos [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

Actividades  Terminal  16 de nov 17:30
alexgb@alexgb-examen: ~/Descargas
alexgb@alexgb-examen:~/Descargas$ md5sum pro-elements.zip
2e8a6f0b6548ce8123989da1f852bee7  pro-elements.zip
alexgb@alexgb-examen:~/Descargas$ sha256sum pro-elements.zip
408f4541206793f49966ecb920a3b052bc0d9f7ea5766e741899f690d36cc851  pro-elements.zip
alexgb@alexgb-examen:~/Descargas$
```

Comandos utilizados:

md5sum pro-elements.zip

sha256sum pro-elements.zip

Para utilizar estos comandos para comprobar un archivo descargado de internet tendríamos que tener una copia de este que sepamos que es de una **fuentes confiable**, por ejemplo descargada de la web del desarrollador.

3. Comparación de Algoritmos (análisis):

- Investiga y responde: ¿Por qué los algoritmos **MD5** y **SHA-1** ya no son recomendados para aplicaciones críticas? Da un ejemplo de una situación en la que el uso de estos algoritmos podría representar un riesgo.
- Indica en qué situaciones podría ser aceptable utilizar **MD5** en lugar de algoritmos más seguros como **SHA-256** o **SHA-512**.

Los algoritmos MD5 y SHA-1 ya no son recomendados para aplicaciones críticas debido a sus vulnerabilidades a ataques de colisión, donde dos entradas diferentes pueden producir el mismo hash. Esto permite a un atacante crear un archivo malicioso que tiene el mismo hash que un archivo legítimo, comprometiendo la integridad de los datos.

Razones por las que MD5 y SHA-1 no son recomendados:

- **Vulnerabilidades de colisión:**
 - Se han descubierto métodos para generar colisiones en ambos algoritmos, lo que significa que es posible encontrar dos entradas diferentes que produzcan el mismo hash.
- **Debilidades criptográficas:**
 - Con el avance de la tecnología y la computación, se han desarrollado técnicas que permiten romper la seguridad de estos algoritmos más fácilmente, haciéndolos inadecuados para proteger datos sensibles.
- **Recomendaciones de estándares:**
 - Organizaciones como el Instituto Nacional de Estándares y Tecnología (NIST) han desaconsejado el uso de MD5 y SHA-1 en favor de algoritmos más seguros como SHA-256 y SHA-512.

Ejemplo de riesgo:

Situación de riesgo:

- Imagina un sistema de gestión de contraseñas que utiliza MD5 para almacenar los hashes de las contraseñas de los usuarios. Un atacante podría generar un hash de una contraseña común y compararlo con los hashes almacenados. Si encuentra una coincidencia, podría acceder a la cuenta del usuario, comprometiendo la seguridad del sistema.

Situaciones aceptables para usar MD5:

Verificación de integridad de archivos:

- MD5 puede ser aceptable para verificar la integridad de archivos no críticos, como en la validación de descargas de archivos donde la seguridad no es primordial. Por ejemplo, se puede usar para comprobar que un archivo descargado no está corrupto.

4. Informe final:

- Elabora un informe en un documento de texto (.docx o .pdf) en el que incluyas las respuestas a las preguntas anteriores, capturas de pantalla de la ejecución de los comandos y una breve reflexión sobre la importancia de las funciones hash en la seguridad informática.

Las funciones hash son herramientas clave en la seguridad de la información, ya que nos ayudan a asegurar que los datos que manejamos se mantengan intactos y auténticos. Al generar un valor único a partir de un conjunto de datos, nos permiten verificar rápidamente si algo ha cambiado. Esto es especialmente importante en situaciones como la firma digital y la protección de contraseñas.

Sin embargo, no todos los algoritmos de hash son igual de seguros. Algunos, como MD5 y SHA-1, han demostrado ser vulnerables a ataques, lo que puede poner en riesgo la información sensible. Por eso, es fundamental optar por algoritmos más robustos, como SHA-256 o SHA-512, que ofrecen una mayor protección.